Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of the Claims:

1. (Previously Presented) A programmable processor for executing a plurality of programs, said programmable processor comprising:

an execution pipeline having a depth of a plurality of execution stages and a depth less than or equal to said plurality of programs wherein each of the plurality of programs comprises a plurality of instructions and having an average pipeline latency of one instruction per cycle; and

an interleaver for interleaving instructions from said plurality of programs and providing said instructions to said pipeline for execution such that the number of said plurality of programs that are interleaved is greater than or equal to the depth of the pipeline.

2. (Previously Presented) The processor of claim 1 or claim 23 wherein said pipeline has a datapath with a depth equal to said number of programs.

3. (Previously Presented) The processor of claim 1 wherein a next instruction from one of said plurality of programs is not provided to said pipeline until a previous instruction of said one of said plurality of programs has completed and in the meantime an instruction from another program is being executed by said pipeline.

4. (Previously Presented) The processor of claim 1 or claim 23 wherein each program of said plurality of programs is independent of the other of said plurality of programs.

5. (Previously Presented) The processor of claim 1 or claim 23 further including an output buffer for storing out of order data output.

6. (Previously Presented) The processor of claim 1 or claim 23 further including one or more of a register copy, program counter, and program counter stack provided for each of said plurality of programs.

7. (Previously Presented) The processor of claim 6 wherein one or more of control and computing resources, instructions, instruction memory, data paths, data memory, and caches are shared by said plurality of programs.

8. (Previously Presented) The processor of claim 1 or claim 23 wherein said processor executes SIMD vector instructions of vector length N and executes in parallel a plurality of instructions having SIMD vector lengths that sum up to N.

9. (Previously Presented) The processor of claim 1 or claim 23 wherein said instructions comprise load instructions for loading data from a data memory.

10. (Previously Presented) The processor of claim 1 or claim 23 wherein said instructions comprise store instructions for storing data in a memory.

11. (Previously Presented) The processor of claim 9 wherein said data memory comprises a cache.

12. (Previously Presented) The processor of claim 9 wherein address space of said data memory comprises a frame buffer unit.

13. (Previously Presented) The processor of claim 9 wherein address space of said data memory comprises a texture memory unit.

14. (Previously Presented) A method of executing instructions from a plurality of programs comprising:

identifying N programs of said plurality of programs wherein each of the plurality of programs comprises a plurality of instructions;

interleaving instructions from said N programs in a processor pipeline wherein said pipeline has an average latency of one instruction per cycle and wherein said pipeline has a depth of a plurality of execution stages; and

executing said instructions such that a first instruction from one of said N programs is completed before beginning execution of a second instruction of said one of said N programs wherein no no-op or idle is inserted into the pipeline for the purpose of ensuring that said first instruction is completed before beginning execution of said second instruction and wherein an instruction from another of said N programs has been interleaved while said first instruction is executing.

15. (Previously presented) The method of claim 14 further including the step of assigning a program counter to each of said N programs.

16. (Previously presented) The method of claim 14 further including the step of assigning a register to each of said N programs.

17. (Previously presented) The method of claim 14 wherein said processor pipeline has a depth of N.

18. (Previously presented) The method of claim 14 wherein said processor executes SIMD vector instructions of vector length N and executes in parallel a plurality of instructions having SIMD vector lengths that sum up to N.

19. – 22. (Canceled)

23. (Previously presented) A programmable processor for executing a plurality of programs, said programmable processor comprising:

an execution pipeline having an average pipeline latency of one instruction per cycle and having a depth of a plurality of execution stages and a depth less than or equal to the plurality of programs wherein each of the plurality of programs comprises a plurality of instructions; and

an interleaver for interleaving instructions from said plurality of programs and providing said instructions to said pipeline for execution wherein a next instruction from one of said plurality of programs is not provided to said pipeline until a previous instruction of said one of said plurality of programs has completed and wherein no no-op is inserted into the pipeline for the purpose of ensuring that said next instruction is not provided to said pipeline until said

previous instruction has completed and wherein an instruction from another of said N programs has been interleaved while said first instruction is executing.

24. (Canceled)

25. (Previously Presented) A programmable processor for executing a plurality of programs, said programmable processor comprising:

a target program counter coupled to a plurality of program counters, said target program counter for determining a number of programs to interleave from a plurality of programs that is greater than the number of program counters;

each of said plurality of program counters coupled to an instruction memory;

instructions from said instruction memory coupled to an instruction decode;

said decode coupled to a plurality of registers;

each of said plurality of registers coupled to an operand route;

said operand route coupled to an arithmetic datapath;

said datapath and an output of a data memory coupled to a result route; and

an output of said result route fed back to each of said plurality of registers.

26. (Previously Presented) The programmable processor of claim 25 wherein said plurality of program counters is equal to said plurality of programs to be interleaved.

27. (Previously Presented) The programmable processor of claim 25 wherein said plurality of registers is equal to said plurality of programs to be interleaved.

28. (Previously Presented) The programmable processor of claim 25 wherein said plurality of registers is more than said plurality of programs to be interleaved.

29. (Previously Presented) The programmable processor of claim 25 wherein said instruction memory is larger than needed to hold said plurality of programs.

30. (Previously Presented) The programmable processor of claim 25 wherein said data memory is larger than needed to hold a data set accessed by each of said plurality of programs.

31. (Previously Presented) The programmable processor of claim 25 wherein each of said plurality of registers is double buffered and contains twice as many copies of registers as said plurality of programs.

32. (Canceled)

33. (Currently amended) A method, by a programmable processor, of executing one or more instructions from a plurality of programs, comprising:

assigning a first output register slot to a first of said plurality of programs wherein each of the plurality of programs comprises a plurality of instructions;

executing said one or more instructions of said first program until said first program is completed;

loading output of said first program into its reserved space when said first program is completed;

checking to see if all of said plurality of programs are completed;

checking to see if a second output register slot is available to assign to a second program from said plurality of programs when said first program is completed;

checking to see if one or more instructions are available when at least one of said plurality of programs is not completed; and

placing an no-op when no more instructions are available or said second output register slot is not available.